



Université de Caen
IUT Grand Ouest
Sciences des Données
1ère année de STID

SAE 2-01

Conception et Implémentation d'une base de données

CAMARA Mohamed, RENOULT Julien,
RUKIRAMARIGO Axelle, THOMAS Lou-Anne

Année universitaire : 2022 / 2023

Table des matières

Introduction	4
1 Problématique et présentation du projet	5
2 Organisation du projet	6
2.1 Architecture du projet	6
2.2 Création de l'UML :	6
2.3 Répartition des tâches :	7
3 Création de la base de données	8
3.1 Création des tables	8
3.2 Insertion des données	9
3.2.1 Lecture des fichiers	9
3.2.2 Insertion des données au sein des tables	10
4 Développement de l'outil de commande	11
4.1 Développement des fonctions de l'outil de commande	11
4.1.1 Rechercher un produit dans le stock	11
4.1.2 Echanger un produit entre le stock et le rayon	12
4.1.3 Changer de place un produit avec un autre	12
4.1.4 Ajouter de nouveaux stocks de produits et supprimer définitivement des exemplaires	12
4.2 Création des fichiers log	13
4.2.1 Fichier log concernant la gestion des stocks	13
4.2.2 Fichier log concernant les ventes	13
4.3 Distribution automatique des produits dans le rayon	14
5 Interface graphique	15
5.1 Présentation des objectifs	15
5.2 Présentation de l'interface graphique	15
6 Création de l'outil d'analyse des ventes	17
Conclusion	23

Introduction

Le présent rapport a pour objectif de présenter le système de gestion de bases de données (SGBD) que nous avons développé de A à Z dans le cadre de notre projet. Elle présentera aussi le développement de plusieurs outils comme la gestion des stocks et du rayon, la vente des articles et l'analyse des ventes.

Le rapport décrira en détail ces différents processus de conception et de développement mis en œuvre pour créer un système de gestion de bases de données.

Dans l'ensemble, ce rapport offre une vue d'ensemble complète du système de gestion de bases de données que nous avons conçu et mis en place.

1 Problématique et présentation du projet

Suite à une coupure d'internet, l'épicerie de la ville Wulverdinghe n'est plus en mesure d'utiliser les outils de la centrale de Paris. Le gérant fait donc appelle à nos services afin de mettre en place une nouvelle base de données permettant de répondre à ses besoins.

Le rapport est structuré en quatre parties. En premier lieu, il aborde l'organisation du projet, puis il présente la conception de la base de données ainsi que celle de l'outil de commande. Enfin, il propose un outil d'analyse des données.

Données fournies :

1. Inventaire

- a) Deux fichiers JSON rassemblant les inventaires des rayons : **crééale** et **librairie**.
- b) Trois fichiers CSV rassemblant les inventaires des rayons : **viande**, **poisson**, **surgelé**.
- c) Deux fichiers TXT rassemblant les inventaires des rayons : **fruit et légume** et **divers**.

2. Une interface graphique

Ces fichiers sont disponibles en **Annexe 1** de la partie Annexe de ce rapport.

Résultats attendus :

- 1. Une base de données documentée et présentée au format ULM.
- 2. Un outil en ligne de commande pour que les utilisateurs puissent gérer leur stock.
- 3. Adapter l'interface graphique au besoin du gérant.
- 4. Un outil permettant d'analyser les ventes effectuées.

2 Organisation du projet

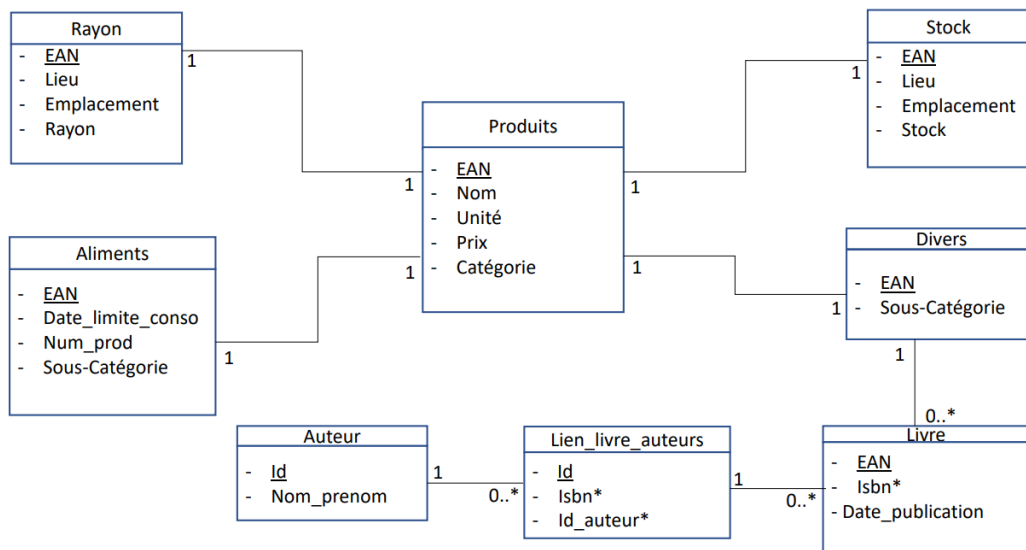
2.1 Architecture du projet

Projet

- > Code
 - > Analyse_ventes
 - > Base_données
 - > DATA
 - > CSV
 - > JSON
 - > TXT
 - > Package
 - > Ticket_caisse
- > Rapport
- > README.txt

2.2 Création de l'UML :

Avant de commencer la conception de la base de données, nous avons étudié les fichiers à notre disposition afin de comprendre l'organisation de l'épicerie et d'élaborer une structure adaptée dans le futur. Nous avons ensuite réalisé l'UML qui formalise notre future base de données :



UML réalisé

L'UML de la base de données "**base.db**" se répartit de la façon suivante, il possède une table **Produits** constituée d'informations générales liées à chaque produit. Nous avons ensuite ajouté une colonne "Catégorie" afin de trier les produits en deux catégories : "Alimentaire" ou "Divers" (produits non alimentaire). De plus, cette table est liée à des tables comportant des informations plus spécifiques à chacun des produits selon leur catégorie : une table **Aliments** et une table **Divers**.

La table **Divers** comporte également des informations en lien avec la table **Livre** répartie en trois tables (Livre, Lien_livre_auteurs, Auteur).

Nous avons ensuite créés une table **Rayon** comportant l'emplacement des produits dans le magasin et une table **Stock** comportant les produits placés en stock.

Nous avons également créée une seconde base donnée intitulé "**analye.db**". Cette base contiendra une unique table "ventes" qui récapitulera toutes les ventes effectuées par l'épicerie.

2.3 Répartition des tâches :

Ce projet se compose ainsi de quatre étapes.

1. Étape 1 : Dans un premier temps, nous allons créer nos tables SQL et y insérer les données contenues dans nos fichiers. Ainsi, nous allons récupérer les données de nos différents fichiers CSV, JSON et TXT pour en faire des listes que nous allons insérer dans nos tables.
2. Étape 2 : La seconde étape concerne l'élaboration d'un outil de commande. Pour ce faire, nous avons développé des fonctions pour permettre à un utilisateur quelconque de gérer ses stocks en ligne de commande. Des fichiers logs retraceront l'ensemble des actions et résultats effectués lors de l'utilisation de cet outil.
3. Étape 3 : En troisième lieu, nous nous concentrerons sur le développement des tickets de ventes. Nous développerons ainsi l'interface graphique déjà fournie afin qu'elle puisse être utilisée pour passer les commandes et éditer un ticket de vente de la commande.
4. Étape 4 : Enfin, nous développerons ici un outil d'analyse qui permettra au gérant d'étudier la performance de ses ventes via l'utilisation du logiciel Rstudio.

3 Création de la base de données

3.1 Création des tables

Tout d'abord, nous avons créé une base de données en **Sqlite** intitulé : **base.db**. Nous avons ainsi écrit l'ensemble de nos requêtes en respectant l'UML créé précédemment. Il est important de noter que nous avons créés des lieux de stockages présents dans la colonne **lieu** de la table **Stock**.

Ces **lieux** sont répartis de la façon suivante :

1. **Stock A** : Viandes et Poisson ;
2. **Stock B** : Surgelé ;
3. **Stock C** : Céréale ;
4. **Stock D** : Divers ;
5. **Stock E** : Fruit et Légume.

Concernant les emplacements des produits dans chacun de ces lieux, nous les avons générés arbitrairement en assignant un numéro allant de 1 au nombre de produits présents dans le lieu de stock. Nous avons fait de même pour l'emplacement des produits en rayon.

Les rayons sont répartis de la façon suivante :

1. **Frais** (Viandes, Poissons et Surgelé) ;
2. **Fruits_Légumes** ;
3. **Céréales** ;
4. **Librairie** ;
5. **Général** (divers).

3.2 Insertion des données

3.2.1 Lecture des fichiers

Tout d'abord, avant de s'occuper de l'insertion des données, nous devons récupérer les données des fichiers fournis. Pour ce faire, nous avons lu chacun des fichiers et récupéré les données sous forme de liste.

Pour nous faciliter la manipulation de ces listes (recensant les données contenues dans chacun des fichiers), nous avons développé une fonction intitulée **produits_inf** permettant de rechercher des informations dans nos listes de données. Cette fonction se base sur un dictionnaire intitulé **Name_colonne**, le nom du produit et les éléments à récupérer. **name_colonne** est donc un dictionnaire qui est utilisé pour stocker les noms des colonnes correspondant à chaque liste de données créée.

Ainsi, nous avons commencé par la lecture des fichiers de données aux formats CSV, TXT et JSON. Pour chacun des fichiers, nous procédions de la façon suivante :

1. On lit et récupère les données de chaque fichier dans des listes.
2. On ajoute les noms des colonnes à notre dictionnaire `name_colonne`.
3. On supprime le nom des colonnes présent dans notre liste recensant les informations lues.

Ensuite, nous procédions à la préparation des données à insérer dans chaque table. Cela implique de réaliser différentes opérations pour récupérer des colonnes spécifiques pour chaque table en combinant plusieurs listes. De plus, nous avons créé les emplacements de stockage dans la table "stock". Pour ce faire, nous avons parcouru notre liste et attribué le numéro d'emplacement correspondant à la position de chaque élément dans cette liste.

3.2.2 Insertion des données au sein des tables

Nous avons commencé par insérer des données dans la catégorie "Viande et Poisson" en utilisant une boucle for et la fonction enumerate pour parcourir la liste `lp.stock_poisson_viande`. Pour chaque élément de la liste, la fonction `ps.sqliteinsertdata` est appelée pour insérer les données dans la base de données.

Ensuite, il y a des sections similaires pour les catégories "Surgelé", "Céréale", "Divers", "Livres" et "Fruits et légumes". Chaque section utilise une boucle for et la méthode enumerate pour parcourir les listes correspondantes et insérer les données dans la base de données.

Après cela, nous passons maintenant à l'insertion de données dans la table "Rayon". Il y a des sections pour les rayons "Frais", "Fruits_légumes", "Céréale", "Librairie" et "Générale". Les boucles for et la méthode enumerate sont utilisées pour parcourir les listes correspondantes et insérer les données dans la base.

Ensuite, il y a une section pour l'insertion de données dans la table "Livre". Nous avons utilisé une boucle for pour parcourir la liste `lp.Livres_inf`. Pour chaque élément de la liste, une conversion de date est effectuée, puis la fonction `ps.sqliteinsertdata` est appelée pour insérer les données dans la base de données.

Enfin, il y a des sections pour l'insertion de données dans les tables "Auteur" et "Lien_Livre_Auteur". Les boucles for sont utilisées pour parcourir les données correspondantes. Nous avons effectué des manipulations sur la clé keys pour éliminer les guillemets, puis utilise la fonction `ps.sqliteinsertdata` pour insérer les données dans les bases de données respectives.

Ces différentes sections d'insertion de données utilisent la fonction `ps.sqliteinsertdata` avec différents arguments pour spécifier la base, la requête SQL et les données à insérer. Le code contient également des lignes commentées qui appellent la fonction `ps.sqlitedonnee` pour récupérer et afficher des données à partir de la base de données. Nous avons donc effectué des insertions de données dans plusieurs tables d'une base de données en utilisant des boucles et des requêtes SQL. Les données sont extraites à partir de différentes listes et insérées dans les tables appropriées

4 Développement de l'outil de commande

Dans cette partie, nous avons développé un outil en ligne de commande afin que les utilisateurs puissent gérer leur stock et effectuer les opérations suivantes :

- Rechercher un produit dans le stock
- Retirer un produit du stock
- Ranger un produit
- Rentrer un produit dans le stock

Un fichier log retraceront toutes les actions produites par l'utilisateur : un fichier nommé `resumer_des_gestions_Stock_Rayon.log`.

4.1 Développement des fonctions de l'outil de commande

Nous avons tout d'abord commencé par écrire les fonctions pour effectuer les opérations liées à la gestion d'un inventaire de produits. Par la suite, nous avons rassemblé toutes ces fonctions dans un même script nommé **Gérer_stock.py**. Ainsi, l'outil de commande est développé comme suit :

À chaque itération de la boucle, l'utilisateur est invité à saisir une commande parmi les options disponibles : changer de place un produit (C), rechercher un produit (R), échanger un nombre d'exemplaires d'un produit (E), ajouter de nouveaux stocks de produits (A) ou retirer définitivement des exemplaires d'un produit (S).

Une fois la commande saisie, le script vérifie si elle fait partie des commandes valides. Si c'est le cas, il demande des informations supplémentaires à l'utilisateur en fonction de la commande choisie, telles que le nom du produit, le type de lieu (stock ou rayon) et éventuellement le lieu ou l'emplacement concerné.

Enfin, l'utilisateur est invité à décider s'il souhaite continuer ou arrêter la gestion des commandes. Si la réponse est "N", la boucle se termine et le script s'arrête. Sinon, la boucle continue à la prochaine itération.

4.1.1 Rechercher un produit dans le stock

Cette fonction peut être utilisée en rentrant **R** dans l'outil de commande.

Nous avons ici cherché à créer une fonction permettant à l'utilisateur de rechercher un produit en stock ou en rayon en fonction de son nom. Ainsi, la fonction **recherche_prod** effectue une requête SQL sur la base de données et renvoi l'EAN du produit, le lieu où se trouve le produit, l'emplacement du produit et le nombre d'exemplaires disponibles.

Si le produit est trouvé mais qu'il n'a pas d'exemplaires disponibles, la fonction renvoi les mêmes informations et prévient l'utilisateur que le produit n'est plus dans le lieu de stockage

ou à sa place en rayon.

En cas d'erreur critique dans la recherche, la fonction renvoie une chaîne de caractères indiquant "ERREUR CRITIQUE AU NIVEAU DE LA RECHERCHE".

4.1.2 Echanger un produit entre le stock et le rayon

Cette fonction peut être utilisée en rentrant **E** dans l'outil de commande.

La fonction **echange_produit** réalise un échange de produits entre le stock et le rayon, mettant à jour les quantités dans la base de données, tout en gérant les cas de stock insuffisant.

Plus précisément : La fonction vérifie également le nombre de produits demandés, s'assurant qu'il est positif et entier. Si la vérification est validée la fonction procède à l'échange en mettant à jour les quantités en stock et/ou en rayon dans la base de données.

La fonction prend également en compte si le nombre de produits dans le stock est insuffisant. En effet, si le nombre de produits dans le stock est insuffisant, elle propose à l'utilisateur de retirer les produits restants malgré tout. Si l'utilisateur accepte, les produits restants sont retirés du stock et ajoutés au rayon et, si l'utilisateur refuse, l'échange est annulé. En cas de réponse incorrecte de la part de l'utilisateur ou si le produit n'est plus disponible dans le lieu d'échange (Stock ou Rayon) , une erreur est signalée.

4.1.3 Changer de place un produit avec un autre

Cette fonction peut être utilisée en rentrant **C** dans l'outil de commande.

La fonction **echange_place_stock_rayon** permet de déplacer un produit d'un emplacement à un autre dans le stock ou le rayon. Pour rappel, nous avons cinq stock différents (A,B,C,D,E) et cinq rayons différents (frais, fruits et légumes, céréales, librairie et générale). L'objectif de cette fonction est donc de mettre à jour le déplacement d'un produit d'un emplacement à un autre dans la table stock et rayon.

4.1.4 Ajouter de nouveaux stocks de produits et supprimer définitivement des exemplaires

Cette fonction peut être utilisée en rentrant **A** pour ajouter des produits et **S** pour en supprimer définitivement à partir de l'outil de commande.

La fonction **ret_aj_def** permet donc d'ajouter ou de supprimer définitivement un produits. Ains, si le nombre de produits disponibles ajouté au nombre de produit à ajouter et à retirer est supérieur ou égal à zéro, la fonction met à jour la base de données en ajoutant ou retirant le nombre de produits spécifié. Elle renvoie les informations du produit avant et après la modification.

Si le nombre de produits disponibles est supérieur à zéro mais insuffisant pour satisfaire la demande, la fonction demande à l'utilisateur s'il souhaite retirer les produits restants. Si l'utilisateur répond "OUI", les produits restants sont retirés de la base de données et les informations avant et après la modification sont renvoyées. Si l'utilisateur répond "NON", l'opération est annulée et un message correspondant est renvoyé.

4.2 Création des fichiers log

Les fichiers logs présentés ci-dessous ont été mis en place afin de pouvoir suivre l'exécution de nos programme. Il permettent de suivre les actions réalisées par le programme et de signaler les erreurs rencontré.

4.2.1 Fichier log concernant la gestion des stocks

Le fichier **resumer_des_gestions_Stock_Rayon.log** retrace un résumé des actions effectuées lors de la gestion des stocks et des rayons. Les actions telles que la recherche d'un produit, l'échange d'exemplaires entre le stock et le rayon ou encore le changement de place d'un produit sont enregistrées dans ce fichier. Il est utilisé pour suivre les opérations effectuées sur les stocks et les rayons, fournissant ainsi un historique des actions passées. Ce fichier est créé lorsque le programme est lancé pour la gestion des stocks et des rayons.

```
2023-06-08 12:11:09,434 - WARNING - ATTENTION : ActiveMQ in Action : Recherche RAYON : Lieu : Librairie, Emplacement : 233, Exemple  
nulle : 0  
2023-06-08 12:11:09,454 - INFO - ActiveMQ in Action : STOCK : 3 -> 2  
2023-06-08 12:11:09,454 - INFO - ActiveMQ in Action : RAYON : 0 -> 1  
2023-06-08 12:11:09,454 - WARNING - ATTENTION : Stylo pointe moyenne : Recherche RAYON : Lieu : Général, Emplacement : 25, Exemple  
nulle : 0  
2023-06-08 12:11:09,476 - INFO - Stylo pointe moyenne : STOCK : 12 -> 6  
2023-06-08 12:11:09,476 - INFO - Stylo pointe moyenne : RAYON : 0 -> 6
```

Extrait du fichier **resumer_des_gestions_Stock_Rayon.log**

4.2.2 Fichier log concernant les ventes

Le fichier **resumer_des_ventes.log** retrace un résumé des actions effectuées lors des ventes de produits. Les actions telles que la recherche d'un produit avant une vente, l'enregistrement des ventes ou encore les mises à jour des stocks après une vente sont enregistrées dans ce fichier. Il est utilisé pour suivre les opérations effectuées lors des ventes et pour garder une trace des ventes précédentes. Ce fichier est créé lorsque le programme est lancé pour la gestion des ventes.

```
2023-06-08 12:01:45,741 - INFO - Disque dur externe 2,5 Elements 2 To : RETIRER RAYON : 8 -> 7  
2023-06-08 12:01:46,608 - INFO - Disque dur externe 2,5 Elements 2 To : RETIRER RAYON : 7 -> 6
```

Extrait du fichier **resumer_des_ventes.log**

4.3 Distribution automatique des produits dans le rayon

Afin d'insérer des produits en rayon nous avons créé un programme pour répartir de façon automatique les différents produits. Pour cela, nous avons mis la moitié des produits en rayon, si le nombre d'exemplaires était impair, nous avons arrondi au nombre inférieur.

5 Interface graphique

5.1 Présentation des objectifs

Le script **VenteGui.py** crée une interface utilisateur permettant à l'utilisateur de saisir des informations dans une barre de recherche. Il comprend également une fonctionnalité de création de ticket de vente avec une zone d'affichage disponible.

Cependant, il reste à compléter le script pour pouvoir enregistrer les articles de vente dans la table « **ventes** » de la base de données « **analyse.db** ». Il faudra également afficher les informations de vente dans le fichier journal « **resumer_des_ventes.log** » pour enregistrer les actions effectuées. De plus, il sera nécessaire de générer un ticket de vente qui enregistre les produits, les quantités, les prix unitaires et le montant total de la vente.

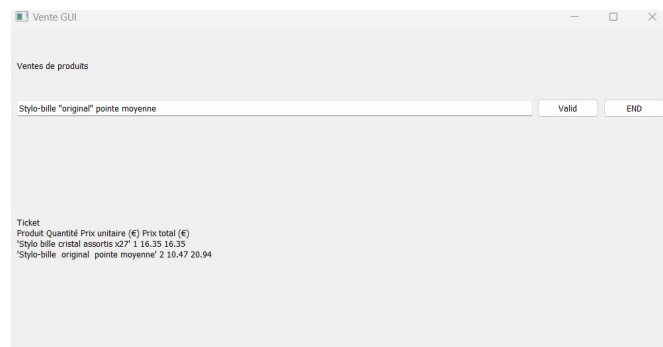
De plus, certaines vérifications devront être mises en place, notamment pour détecter les erreurs de saisie, vérifier la disponibilité des produits en rayon, et annuler la vente en cas de problème. Le script permet également de mettre à jour les quantités disponibles dans les bases de données après chaque vente.

Nous avons complété les fonctionnalités déjà existante. Nous avons permis à un utilisateur d'enregistrer les ventes saisies dans la table « ventes » de la base « analyse.db ». Nous affichons les informations de vente dans le fichier log, nous générons des tickets de vente avec les détails des produits vendus en effectuant les vérifications nécessaires pour mettre à jour les quantités disponibles.


5.2 Présentation de l'interface graphique

L'interface se présente comme suit :

- On y retrouve une barre de recherche pour que l'utilisateur puisse entrer le nom du produit
- Une zone ticket, qui complète le ticket de vente au fur et à mesure
- Un ticket **Valid** pour valider le produit renseigné et un ticket **End** pour finaliser la vente et générer le ticket



Interface utilisateur

 Ticket de caisse ✕

Ticket

Produit	Quantité	Prix unitaire (€)	Prix total (€)
'Stylo bille cristal assortis x27'	1	16.35	16.35
'Stylo-bille original pointe moyenne'	2	10.47	20.94

Total à payer (€) : 37.29

09-06-2023 11 heures 08 minutes 20 secondes

Ticket de caisse généré

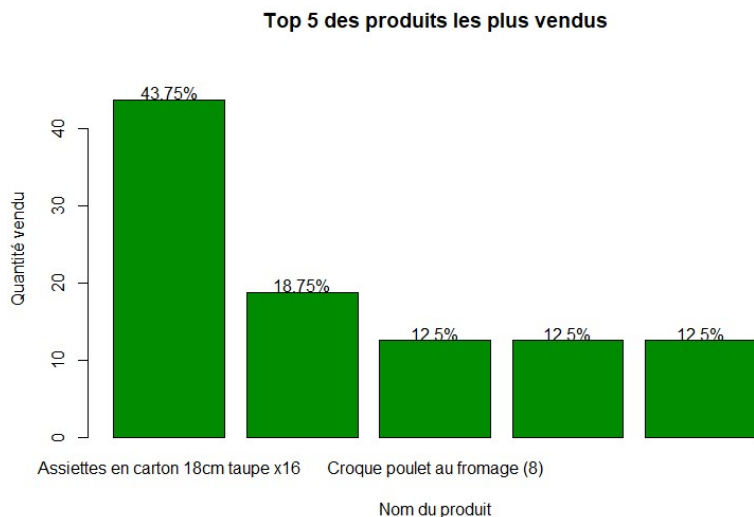
Les ticket générés sont disponible au format txt dans le dossier **Ticket_caisse**.

6 Création de l'outil d'analyse des ventes

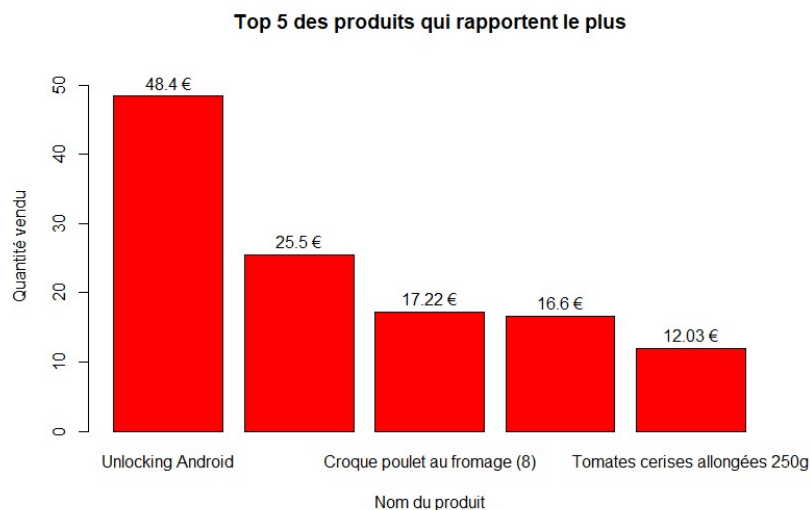
Afin de permettre à l'épicerie de pouvoir suivre ses ventes, nous avons utilisé le code fourni pour récupérer les données, créées précédemment, de notre table Ventes situé dans la base de données **analyse.db**. La table comprend des informations essentielles telles que le nom du produit, la quantité vendue, le prix unitaire, le prix total, la date et l'heure de chaque vente. Nous avons ensuite créé un fichier CSV à partir de ces données (cf. **compte_rendu.py**). Pour créer notre outil d'analyse, nous avons décidé d'importer ce fichier CSV dans le logiciel Rstudio afin de créer des analyses automatisées.

Le fichier Rstudio s'organise comme suit :

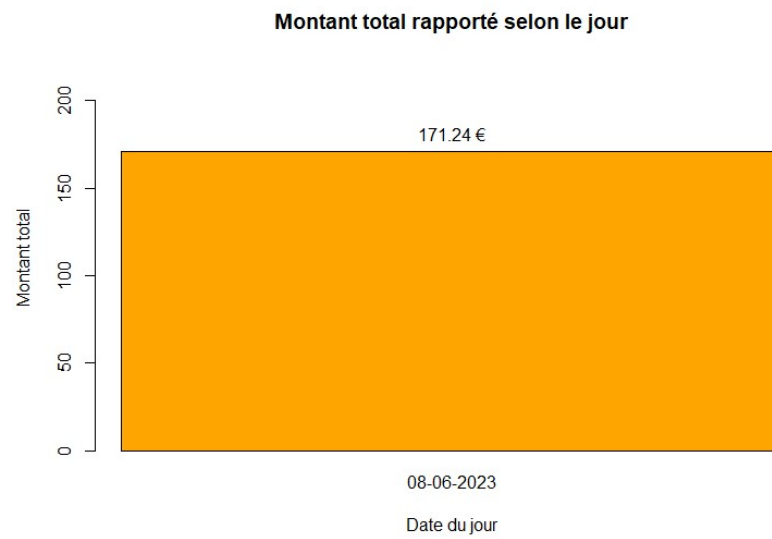
1. Top 5 des produits les plus vendus



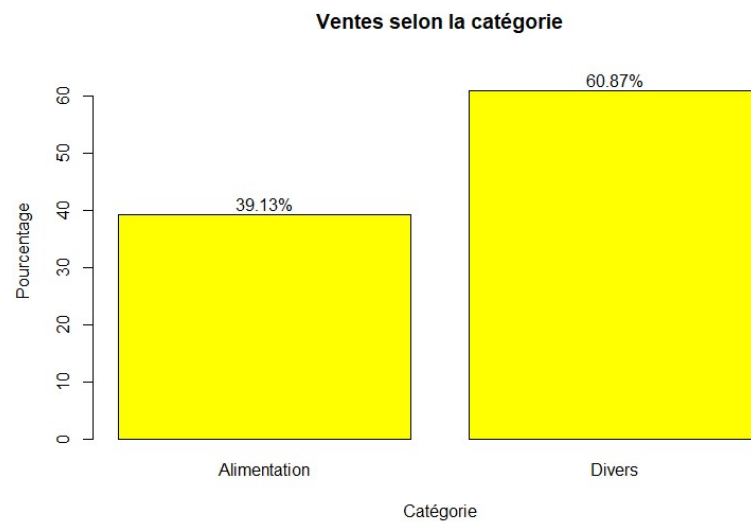
2. Top 5 des produits qui rapportent le plus



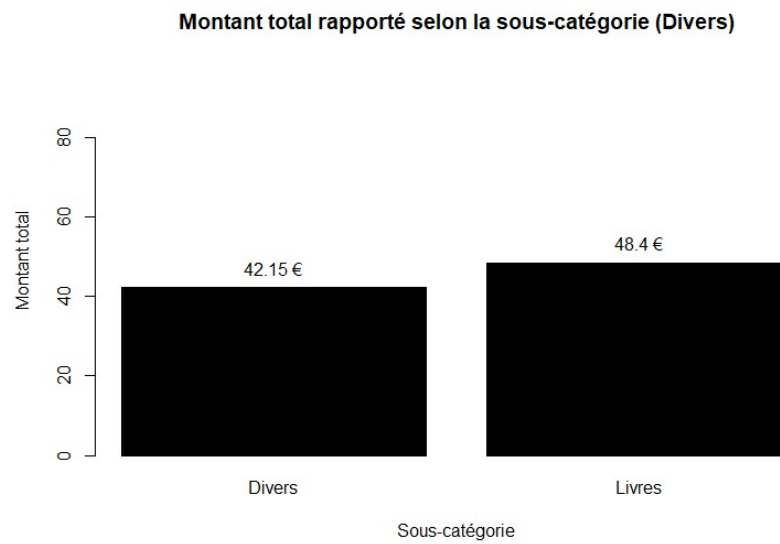
3. Montant total rapporté selon le jour



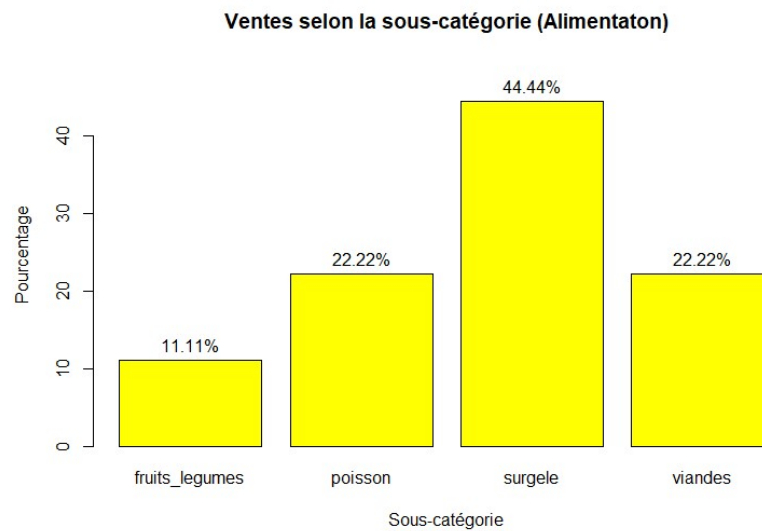
4. Ventes selon la catégorie



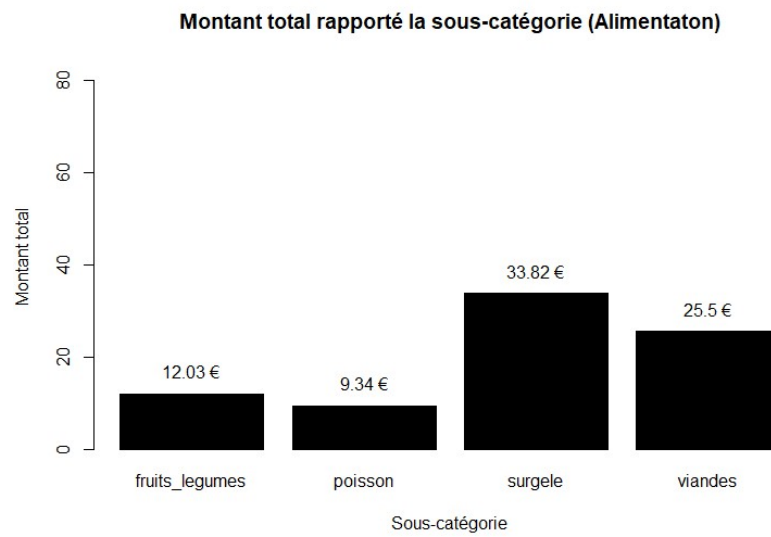
5. Montant total rapporté selon la catégorie (Divers)



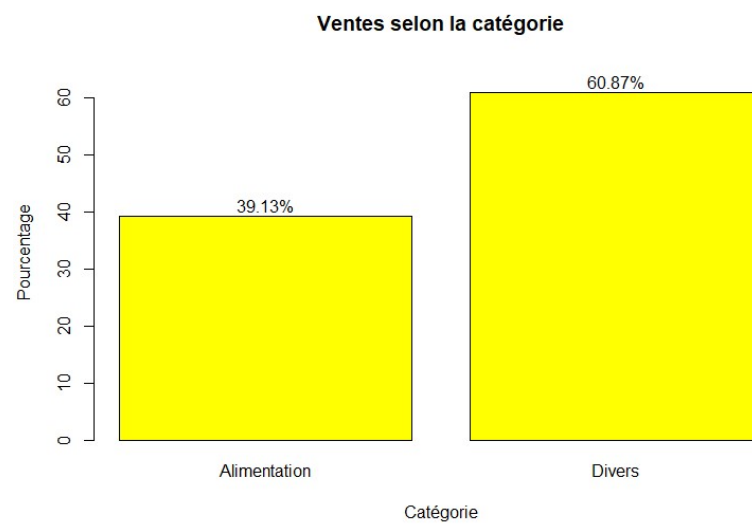
6. Ventes selon la sous-catégorie (Alimentation)



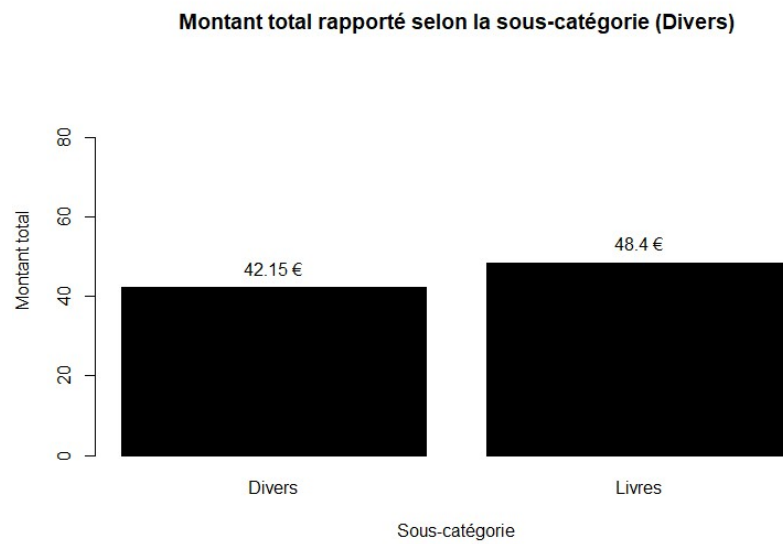
7. Montant total rapporté la sous-catégorie (Alimentation)



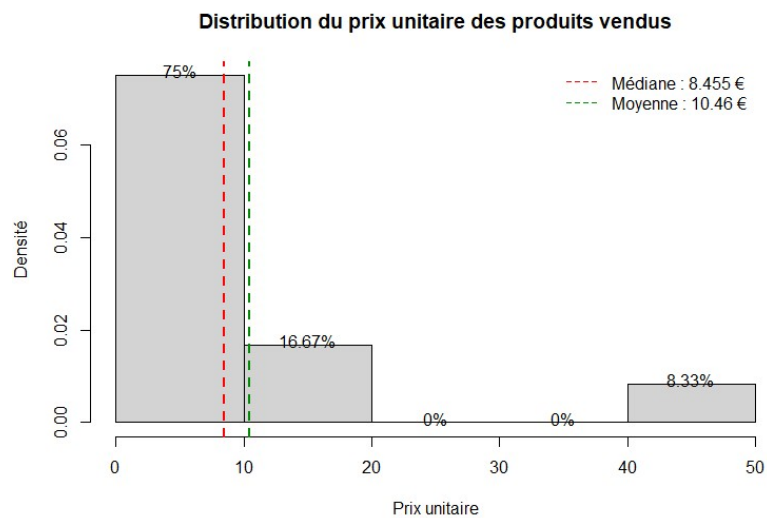
8. Ventes selon la sous-catégorie (Divers)



9. Montant total rapporté selon la sous-catégorie (Divers)



10. Distribution du prix unitaire des produits vendus



Conclusion

Ce projet nous a ainsi permis de mettre en place un système de gestion de bases de données. Nous avons mis en relation chacune de nos tables et avons créé des fonctions qui permettent à l'utilisateur de naviguer dans notre base de données. Nous avons également utilisé Python pour créer une interface graphique permettant à l'utilisateur de saisir les informations de vente et de générer des tickets de caisse détaillés.

Ensuite, nous avons créé un outil d'analyse des ventes en utilisant RStudio. Une fois lancé, cet outil permet à l'utilisateur de pouvoir suivre ces ventes graphiquement afin de visualiser les produits les plus vendus, les produits les plus rentables, le montant total rapporté par jour, les ventes par catégorie et sous-catégorie, ainsi que la distribution des prix unitaires.

Pour finir, ce projet nous a permis de créer un système de gestion de base, se mettant à jour automatiquement si une action se produit et repère l'erreur s'il y a un souci. Elle nous permet de penser davantage à la création de la base, mais aussi comment la gérer pour éviter ainsi des problèmes critiques qu'ils peuvent apparaître. De plus, on devait imaginer la création des différents outils permettant de faire une ou plusieurs actions qui modifie en conséquent la base.

Pour conclure, ce projet était très intéressant dans un point de vu de réflexion dans la création et la mise en place des différents outils pour offrir une gestion de la base la plus propre possible.